PPC440 FP2 Architecture

IBM Corporation

IBM Version 2.0
April 11, 2011 12:01 PM

Change Summary

Table 1. Change Log

Rev	Made On	By	Description
0.1	6/15/01	KAD	Broke out Architecture from FP2 microarchitecture
0.2	7/19/01	KAD	Removed copy CR and copy CI mul-add instructions as they are redundant with CopyARand AI. Added in D-forms of load/store. Removed dual add/sub-across - requires changes that could hurt timing. Removed some multiply and multiply-add variants that did seemed extraneous. Removed gather - requires substantial rip-up. Changed mnemonics and descriptions from dual to parallel. Changed “real” to “primary” & “imaginary” to “secondary” for more generality and less confusion.
0.3	07/26/01	KAD	Extended parallel moves to cover all flavors. Removed D-forms of loads and stores, no primary opcodes available in the Stealth environment. Extended *Cross Mul-add* to cover Cross (neg) multiply-[add | sub] Extended *Cross Copy Mul-add* to include negative-madd and msub. Added *cross copy-secondary multiply* for symmetry. Changed mnemonics so that *parallel* refers only to SIMD instructions whose operands do not change sides, *cross* refers only to SIMD instructions that use data from one register file and write it to the other, and *secondary* refers to non-SIMD instructions that access the secondary register file.

0.4	08/07/01	KAD	Removed everything that was struck-through, including all traces of D-form load/stores. Removed single-precision non-memory instructions. Note that all instructions are handled in DP internally - SP forms only rounded to SP. Added opcode form to instruction tables. Added Floating cross multiply For consistency, I changed the cross multiplies to only vary the A operand (i.e., sel between AP and AS). Cross-complex, however, remains the same.
1.0	08/07/01	KAD	Formatting & editorial changes. Replaced “DS-mode” instructions (primary opcode 1) with “DPS-mode” instructions (primary opcode 4). Removed Chapter 6: “Instruction Descriptions” since all of the instructions are defined unambiguously in Chapter 5: “Instruction Set”.
1.1	08/24/01	KAD	Corrections sent in from our viewers at home: Separated convert and compare; Replaced ~ with -; fixed typos & cut and paste errors; fixed stfpsu, stfss, stfssu descriptions; Added note that DPS-forms inclusion is pending approval of proposed changes to 440. <u>All</u> SP stores that overflow are now forced to infinity.
1.2	08/30/01	KAD	Removed DPS-forms of load and store - the required changes to Avenger (440 in Cu-11) would have too much of an impact on schedule. Renamed fxcxmadd (complex multiply-add) to fxcxnpma. Added fxcxnsma instruction. Combined and moved asymmetric and complex madd instructions to opcode 4 space. Bits 26:27 ==0b11, so no collisions with MAC (== 0b0x) or VMX (=0bx0). Created new sections for instructions - removed unused operands for specific opcode maps.
1.3	08/31/01	KAD	Broke complex instructions out into own sections. Added 2 instructions to the complex section: fxcxma and fxcxnms Moved load and store instructions into allocated space in opcode 31, ‘cause that’s what all the cool kids do.

1.35	09/24/01	KAD	Changed secondary opcodes for ld/st indexed (lfpdx collided with PPC440 instruction dlmzb pri=31 sec=78). Inverted sense of secondary and cross. For the record, other allocated secondary opcodes (under primary 31) snarfed by 440 are: 454, 966, 486, 998
1.4	10/22/01	KAD	Added clarification that instructions which include negative multiply-add type functionality perform rounding before the final negation. This is consistent with the PowerPC defined negative multiply-add type instructions.
1.46	11/13/01	KAD	Fixed move operation diagram - removed FRA operand. Added text explaining that mnemonic formats for ternary operations follow the conventions in PowerPC Book-E. Changed top level drawing to show muxing for load/store.
1.47	03/02/02	KAD	Modified tables to make more clear which signals are active low (the bars don't show up too well in PDF). Updated "conceptual view" to include port names.

Chapter 1. Overview

FP2 is the name of the SIMD-like floating-point instruction extensions to the PowerPC BookE ISA.

FP2 requires registers and instructions specified by the PowerPC BookE Instruction Set Architecture. Therefore, it may only be implemented in a PowerPC BookE environment. In other words, FP2 implementations must include support (hardware or software hooks) for all of the PowerPC BookE floating-point instructions.

Some of the instructions defined by FP2 truly fall into the category of Single Instruction Multiple Data (SIMD), however many of the instructions are not - strictly speaking - SIMD. Some of the instructions will cause two different (yet closely related) operations to be performed. Other instructions will only cause a single operation to occur on a single set of data.

All of the *new* non-memory instructions defined by the FP2 architecture are double-precision operations. This reduces rounding error as well as intermediate overflow or underflow which may occur with single-precision operations. In order to keep the data footprint compact, operands can be loaded in single-precision format. All FP2 single-precision store instructions are defined to convert double precision results to single precision by truncating the mantissa, forcing overflows to infinity, and treating underflow values that can't be represented as denormal numbers to zero. If the application requires that the stored single-precision value be correctly rounded, the parallel round to single precision instruction can be executed before storing.

FP2 does not provide support for enabling exceptions, enabled exception handling, nor status flags. Therefore these instructions are not IEEE-754 compliant. Nevertheless, all results returned conform to the standard’s defined behavior for when exceptions are disabled. Adding support for enabled exceptions would add a great deal of complexity with little gain for the intended target applications.

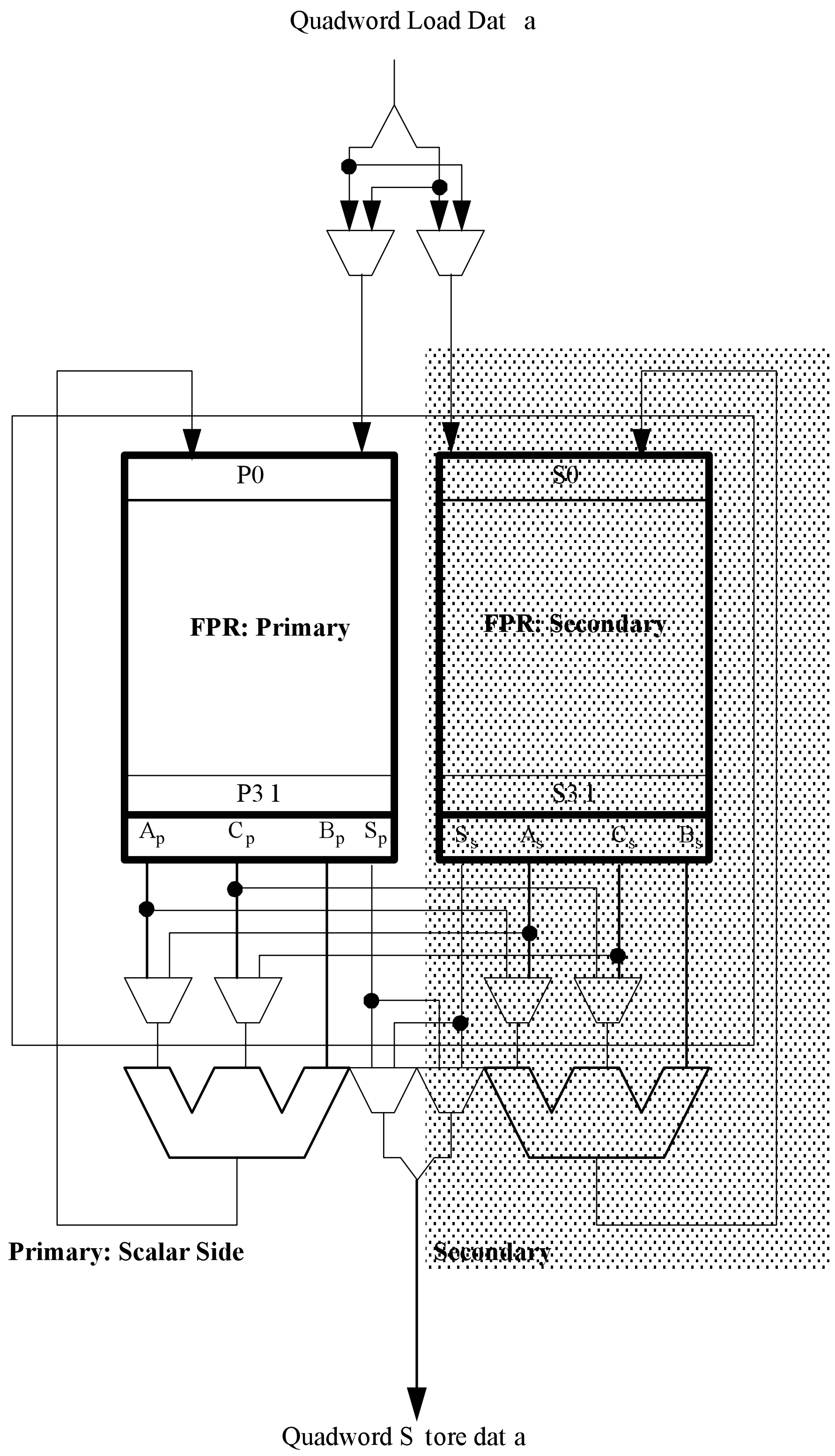


Figure 11.

Conceptual view of an FP2 implementation

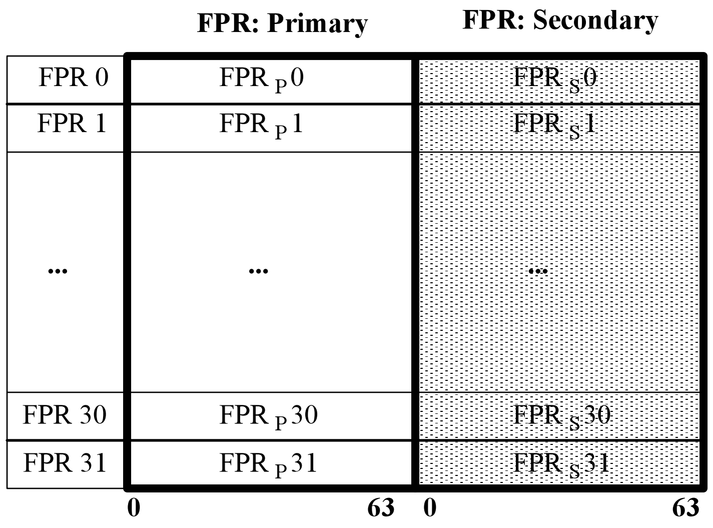
Chapter 2. Registers

2.1 Floating-Point Register File

FP2 instructions operate on data from, and return results to, a dual floating-point register (FPR) file. The dual FPR file is composed of two banks; one is referred to as the primary bank and the other as the secondary bank. The primary bank corresponds with the FPR file defined in the PowerPC BookE Architecture. The secondary bank matches the primary bank in terms of width (64-bits) and depth (32 entries). It is envisioned that the secondary file will often be used to hold the imaginary portion of a complex number pair with the real portion held in the corresponding primary register file. However, the secondary register file's use is not limited to the imaginary component of a complex number; it can just as effectively be used to hold real numbers.

The Secondary FPR file is accessed with the same register addresses as the primary FPR. Thus, operands are typically accessed in pairs: one primary (FPR_P n) and one secondary (FPR_S n). The Secondary FPR is accessed only by the instructions in the FP2 extensions.

Figure 21. Dual FPR File



2.2 Status and Control Register

FP2 does not provide for its own floating-point status and control register (FPSCR). Rather, it uses resources already provided for in the FPSCR described by the PowerPC BookE Architecture. None of the FP2 instructions affect the FPSC; this register is only modified by BookE instructions. However, the FP2 instructions are affected by the mode bits of the FPSCR (that is, rounding and Non-IEEE).

Chapter 3. Exceptions

For all FP2 instructions, floating-point exceptions are treated as disabled regardless of how the FPSCR is set. Nevertheless, an exception will leave its mark on the data. For example, an invalid operation exception on an arithmetic instruction will produce a QNaN which is subsequently propagated throughout later instructions. The table below enumerates the results returned for each of the exceptions. Note that this list also describes the IEEE-754 defined results for disabled exceptions.

Table 31. FP2 results upon one or more exceptions

exception	result
Invalid	arithmetic, round to SP: QNaN compare: unordered convert to int: max signed integer value
Zero Divide	+/- infinity
Overflow	round to nearest: +/- infinity round to zero: +/- largest value round to + infinity: +infinity, - largest number round to - infinity: -infinity, + largest number
Underflow	rounded result (denorm, zero)
Inexact	rounded result

Chapter 4. Storage

4.1 Endianness

Both big and little endian forms are supported. Byte swapping occurs within operand boundaries. Therefore, load/store parallel doubleword would perform swapping for each doubleword, load/store parallel word would perform swapping for each word. Put another way, the actual byte swapping is the same if the data is loaded/stored via multiple single-operand instructions, or via a single multi-operand instruction.

Chapter 5. Instruction Set

Table 51. Relationship of PowerPC instructions to FP2 instructions

class	extensions in Oedipus	PowerPC BookE Mnemonic[1]
add	parallel	fadd, fadds, fsub, fsubs
multiply	parallel, cross	fmul, fmuls
multiply-add	parallel, cross	fmadd, fmadds, fmsub, fmsubs, fnmadd, fnmadds, fnmsub, fnmsubs
divide	none	fdiv, fdivs
estimate	parallel	fres, frsqrte
compare	secondary	fcompo, fcompu
convert to Integer	parallel	fctiw, fctiwz
convert to single precision	parallel	frsp
move	parallel, cross, secondary	fmr, fneg, fabs, fnabs
select	parallel	fsel
move from FPSCR	none	mffs
move to CR from FPSCR	none	mcrfs
move to FPSCR	none	mtfsfi, mtfsf, mtfsb0, mtfsb1
load floating double	parallel, cross, secondary	lfd, lfdx, lfdu, lfdux
load floating single	parallel, cross, secondary	lfs, lfsx, lfsu, lfsux
store floating double	parallel, cross, secondary	stfd, stfdx, stfdu, stfdux

[1]Many of these instructions also have a “dot-form” which causes the CR to be updated with the most significant nibble of the FPSCR.

store floating single	parallel, cross, secondary	stfs, stfsx, stfsu, stfsux
store as integer	parallel	stfiwx
square root	none	fsqrt, fsqrts
double- int to FP	none	fcfid
FP to double-Int	none	fctid

The table above shows which instruction types are further extended in FP2. *Parallel* refers only to SIMD instructions whose operands do not change sides, *cross* refers only to SIMD instructions that use data from one register file and write it to the other, and *secondary* refers to non-SIMD instructions that access the secondary register file (in the case of some move instructions, they also access the primary register file).

All FP2 non-memory-instruction mnemonics begin with “fp”, “fx”, or “fs” while memory-instruction mnemonics begin with “lfp”, “lfx”, “lfs”, “stfp”, “stfx”, or “stfs”.

When the new instruction is the equivalent of two preexisting Book E instructions executed in parallel, the new mnemonic is the same as the preexisting one with “f” replaced with “fp”. Likewise, when the new instructions is the equivalent to a preexisting Book E instruction, but only accesses the secondary register file, the “f” is replaced with “fs”. The two move instructions which move a single element from one register file to the other also begin with “fs”.

New SIMD instructions where operands cross between the primary and secondary sides begin with “fx”, “lfx” or “stfx”, as appropriate.

FP2 instructions are mapped to allocated spaces of Primary Opcode 0, 4, and 31. Furthermore, care was taken to ensure that none of these opcodes step on either the VMX/AltiVec or MAC opcodes.

Mnemonic formats are not explicitly specified here, however they all follow the same rules as preexisting instructions in the PowerPC Book-E architecture. For example, ternary operations are specified as **instr_mnemonic FRT, FRA, FRC, FRB**.

Note: The bit-mapping tables below were added for the convenience of implementors and shall not be provided in any external documentation. These tables indicate a correspondence between secondary opcode bits and features in the design. However, not all combinations of these bits are valid. Only the combinations explicitly defined herein are valid FP2 instructions, all others will result in an illegal instruction exception (unless used by another APU).

5.1 Add Instructions

Table 52. Elementary Arithmetic Instructions (A-Form Primary Opcode 0)

0	0	0	0	0	0	FRT	FRA	FRB	///	X0	/
0					5	6 10	11 15	16 20	21 25	26 30	31

Table 52.
Table 53.

Bit	Description
26	0
27	1
28	1
29	0
30	Subtract/*Add*

Table 53. Elementary Arithmetic instructions XO-field

Table 54. FP2 elementary arithmetic instructions

Instruction	**mnemonic**	**XO**	**description**
Floating Parallel Add	fpadd	12	$A_P + B_P \rightarrow T_P, A_S + B_S \rightarrow T_S$
Floating Parallel Subtract	fpsub	13	$A_P - B_P \rightarrow T_P, A_S - B_S \rightarrow T_S$

5.2 Estimate Instructions

Table 55. Estimate Instructions (A-Form Primary Opcode 0)

0	0	0	0	0	0	FRT	///	FRB	///	X0	/
0					5	6 10	11 15	16 20	21 25	26 30	31

Table 55.
Table 56.

Bit	Description
26	0
27	1
28	1
29	1
30	Recip/*RecipSqrt*

Table 56. Estimate instructions XO-field

Table 57. FP2 estimate instructions

Instruction	mnemonic	XO	description
Floating Parallel Reciprocal Estimate[2]	fpre	14	`RecipEst(`B_P`) ->` T_P`,` `RecipEst(`B_S`) ->` T_S
Floating Parallel Reciprocal Square Root Estimate	fprsqrte	15	`RSqrtEst(`B_P`) ->` T_P`,` `RSqrtEst(`B_S`) ->` T_S

[2]In double precision (Unlike PPC BookE fres)

5.3 Multiply Instructions

Table 58. Multiply Instructions (A-Form Primary Opcode 0)

0	0	0	0	0	0	FRT	FRA	///	FRC	XO	/
0					5	6 10	11 15	16 20	21 25	26 30	31

Table 58.

Table 59.

Bit	Description
26	0
27	1
28	0
29:30	00 - Parallel
	01 - Cross
	10 - Cross copy primary
	11 - Cross copy secondary

Table 59. Multiply instructions XO-field

Table 510. FP2 multiply instructions

Instruction	mnemonic	XO	description
Floating Parallel Multiply	fpmul	8	A_PC_P -> T_P, A_SC_S -> T_S
Floating Cross Multiply	fxmul	9	A_SC_P -> T_P, A_PC_S -> T_S
Floating cross copy-primary Multiply	fxpmul	10	A_PC_P -> T_P, A_PC_S -> T_S
Floating cross copy-secondary Multiply	fxsmul	11	A_SC_P -> T_P, A_SC_S -> T_S

5.4 Multiply-add instructions

Table 511. Multiply-Add Instructions (A-Form Primary Opcode 0)

0	0	0	0	0	0	FRT	FRA	FRB	FRC	X0	/
0					5	6 … 10	11 … 15	16 … 20	21 … 25	26 … 30	31

Table 511.
Table 512.

Bit	Description
26	1
27	Subtract/*Add*
28	Negate
29:30	00 - Parallel
	01 - Cross
	10 - Cross copy primary
	11 - Cross copy secondary

Table 512. Symmetric multiply-add instructions XO-field

Note: The Negate form of multiply-add/sub type instructions perform rounding before the final negation. In other words, this operation is performed like a multiply-add instruction followed by a negate instruction. This is the same behavior as PowerPC defined negative multiply-add/sub type instructions.

Table 513. FP2 symmetric multiply-add instructions

Instruction	**mnemonic**	**XO**	**description**
Floating Parallel Multiply-Add	fpmadd	16	$A_P C_P + B_P$ -> T_P, $A_S C_S + B_S$ -> T_S
Floating Parallel Negative Multiply-Add	fpnmadd	20	$-(A_P C_P + B_P)$ -> T_P, $-(A_S C_S + B_S)$ -> T_S
Floating Parallel Multiply-Subtract	fpmsub	24	$A_P C_P - B_P$ -> T_P, $A_S C_S - B_S$ -> T_S
Floating Parallel negative Multiply-Subtract	fpnmsub	28	$-(A_P C_P - B_P)$ -> T_P, $-(A_S C_S - B_S)$ -> T_S
Floating Cross Multiply-Add	fxmadd	17	$A_S C_P + B_P$ -> T_P, $A_P C_S + B_S$ -> T_S
Floating Cross Negative Multiply-Add	fxnmadd	21	$-(A_S C_P + B_P)$ -> T_P, $-(A_P C_S + B_S)$ -> T_S

Floating Cross Multiply-Subtract	fxmsub	25	$A_SC_P - B_P$ -> T_P, $A_PC_S - B_S$ -> T_S
Floating Cross Negative Multiply-Subtract	fxnmsub	29	$-(A_SC_P - B_P)$ -> T_P, $-(A_PC_S - B_S)$ -> T_S
Floating Cross Copy-Primary Multiply-Add	fxcpmadd	18	$A_PC_P + B_P$ -> T_P, $A_PC_S + B_S$ -> T_S
Floating Cross Copy-Secondary Multiply-Add	fxcsmadd	19	$A_SC_P + B_P$ -> T_P, $A_SC_S + B_S$ -> T_S
Floating Cross Copy-Primary Negative Multiply-Add	fxcpnmadd	22	$-(A_PC_P + B_P)$ -> T_P, $-(A_PC_S + B_S)$ -> T_S
Floating Cross Copy-Secondary Negative Multiply-Add	fxcsnmadd	23	$-(A_SC_P + B_P)$ -> T_P, $-(A_SC_S + B_S)$ -> T_S
Floating Cross Copy-Primary Multiply-Subtract	fxcpmsub	26	$A_PC_P - B_P$ -> T_P, $A_PC_S - B_S$ -> T_S
Floating Cross Copy-Secondary Multiply-Subtract	fxcsmsub	27	$A_SC_P - B_P$ -> T_P, $A_SC_S - B_S$ -> T_S
Floating Cross Copy-Primary Negative Multiply-Subtract	fxcpnmsub	30	$-(A_PC_P - B_P)$ -> T_P, $-(A_PC_S - B_S)$ -> T_S
Floating Cross Copy-Secondary Negative Multiply-Subtract	fxcsnmsub	31	$-(A_SC_P - B_P)$ -> T_P, $-(A_SC_S - B_S)$ -> T_S

5.5 Asymmetric multiply-add instructions

Table 514. Asymmetric multiply-add Instructions (A-Form **Primary Opcode 4**)

0	0	0	1	0	0	FRT	FRA	FRB	FRC	XO	/
0					5	6 10	11 15	16 20	21 25	26 30	31

Table 514.
Table 515.

Bit	Description
26	1
27	1
28	0
29	Negative-Sub Secondary/*Primary*
30	Copy Secondary/*Primary*

Table 515. Asymmetric multiply-add instructions XO-field

Note: The Negate form of multiply-add/sub type instructions perform rounding before the final negation. In other words, this operation is performed like a multiply-add instruction followed by a negate instruction. This holds true even if only the primary or secondary side is negated. This is the same behavior as PowerPC defined negative multiply-add/sub type instructions.

Table 516. FP2 asymmetric multiply-add instructions

Instruction	mnemonic	XO	description
Floating Cross Copy-Primary NSub-Primary Multiply-Add	fxcpnpma	24	$-(A_PC_P - B_P)$ -> T_P, $A_PC_S + B_S$ -> T_S
Floating Cross Copy-Secondary NSub-Primary Multiply-Add	fxcsnpma	25	$-(A_SC_P - B_P)$ -> T_P, $A_SC_S + B_S$ -> T_S
Floating Cross Copy-Primary NSub-Secondary Multiply-Add	fxcpnsma	26	$A_PC_P + B_P$ -> T_P, $-(A_PC_S - B_S)$ -> T_S
Floating Cross Copy-Secondary NSub-Secondary Multiply-Add	fxcsnsma	27	$A_SC_P + B_P$ -> T_P, $-(A_SC_S - B_S)$ -> T_S

5.6 Complex multiply-add instructions

Table 517. Complex multiply-add Instructions (A-Form **Primary Opcode 4**)

0	0	0	1	0	0	FRT	FRA	FRB	FRC	XO	/
0					5	6 10	11 15	16 20	21 25	26 30	31

Table 517.
Table 518.

Bit	Description
26	1
27	1
28	1
29	Negative-Sub Secondary
30	Negative-Sub Primary

Table 518. Complex multiply-add instructions XO-field

Note: The Negate form of multiply-add/sub type instructions perform rounding before the final negation. In other words, this operation is performed like a multiply-add instruction followed by a negate instruction. This holds true even if only the primary or secondary side is negated. This is the same behavior as PowerPC defined negative multiply-add/sub type instructions.

Table 519. FP2 complex multiply-add instructions

Instruction	mnemonic	XO	description
Floating Cross Complex NSub-Primary Multiply-Add	fxcxnpma	29	$-(A_SC_S - B_P) \rightarrow T_P,\ A_SC_P + B_S \rightarrow T_S$
Floating Cross Complex NSub-Secondary Multiply-Add	fxcxnsma	30	$A_SC_S + B_P \rightarrow T_P,\ -(A_SC_P - B_S) \rightarrow T_S$
Floating Cross Complex Multiply-Add	fxcxma	28	$A_SC_S + B_P \rightarrow T_P,\ A_SC_P + B_S \rightarrow T_S$
Floating Cross Complex Negative Multiply-Sub	fxcxnms	31	$-(A_SC_S - B_P) \rightarrow T_P,\ -(A_SC_P - B_S) \rightarrow T_S$

5.7 Select Instruction

Table 520. Select (A-Form Primary Opcode 0)

0	0	0	0	0	0	FRT	FRA	FRB	FRC	XO	/
0					5	6 10	11 15	16 20	21 25	26 30	31

Table 520.
Table 521.

Bit	Description
26	0
27	0
28	1
29	0
30	1

Table 521. Floating-point parallel select XO-field

Table 522. FP2 Select instruction

Instruction	mnemonic	XO	description
Floating Parallel select	fpsel	5	A_P ? C_P : B_P -> T_P, A_S ? C_S : B_S -> T_S

5.8 Convert and Round Instructions

Table 523. Convert (X-Form Primary Opcode 0)

0	0	0	0	0	0	FRT	///	FRB	XO	/
0					5	6 8 9 10	11 15	16 20	21 30	31

Table 523.

Table 524.

Bit	Description
21	Convert to Integer
22	0
23	Round[3]
24	1
25	0
26:30	00000

Table 524. Convert & Round instructions XO-field

Table 525. FP2 Convert & Round instructions (X-form)

Instruction	mnemonic	XO[4]	description
Floating Parallel Convert To Integer Word	fpctiw	576	fctiw (B_P) -> T_P, fctiw (B_S) -> T_S
Floating Parallel Convert To Integer Word and round to zero	fpctiwz	704	fctiwz (B_P) -> T_P, fctiwz (B_S) -> T_S
Floating Parallel Round to Single-Precision	fprsp	192	frsp (B_P) -> T_P, frsp (B_S) -> T_S

[3]Round to SP, or round to Zero in conjuction with Convert to I

[4]not used: [21,23] = 00

5.9 Compare Instruction

Table 526. Compare (X-Form Primary Opcode 0)

0	0	0	0	0	0	BF	//	FRA	FRB	XO	/
0					5	6 8	9 10	11 15	16 20	21 30	31

Table 526.
Table 527.

Bit	Description
21	0
22	1
23	0
24	1
25	0
26:30	00000

Table 527. Compare instruction XO-field

Table 528. FP2 Compare instructions (X-form)

Instruction	mnemonic	XO	description
Floating Secondary Compare[5]	fscmp	320	A_S <> B_S => CR[BF]

[5]Does not modify FPSCR, only 440's CR (therefore, unordered and ordered are the same)

5.10 Move Instructions

Table 529. Move (X-Form Primary Opcode 0)

0	0	0	0	0	0	FRT	///	FRB	XO	/
0					5	6 10	11 15	16 20	21 30	31

Table 529.
Table 530.

Bit	Description
21	Cross
22	Secondary
23	Negate or From/*To*
24	Absolute
25	1
26:30	00000

Table 530. Move instructions XO-field

Table 531. FP2 Move instructions (X-form)

Instruction	mnemonic	XO[6]	description
Floating Parallel Move	fpmr	32	B_P -> T_P, B_S -> T_S
Floating Parallel Negate	fpneg	160	$-B_P$ -> T_P, $-B_S$ -> T_S
Floating Parallel Absolute Value	fpabs	96	$\|B_P\|$ -> T_P, $\|B_S\|$ -> T_S
Floating Parallel Negate Absolute Value	fpnabs	224	$-\|B_P\|$ -> T_P, $-\|B_S\|$ -> T_S
Floating Secondary Move	fsmr	288	B_S -> T_S
Floating Secondary Negate	fsneg	416	$-B_S$ -> T_S
Floating Secondary Absolute Value	fsabs	352	$\|B_S\|$ -> T_S

[6] not used = 864, 992, 608, 672, 736

Floating Secondary Negate Absolute Value	fsnabs	480	$-\|B_S\| \rightarrow T_S$
Floating Cross Move	fxmr	544	$B_P \rightarrow T_S$, $B_S \rightarrow T_P$
Floating Secondary Move From Primary	fsmfp	928	$B_P \rightarrow T_S$
Floating Secondary Move To Primary	fsmtp	800	$B_S \rightarrow T_P$

5.11 Load/Store indexed instructions

Table 532. Load/Store Indexed (X-Form **Primary Opcode 31**)

0	1	1	1	1	1	FRT/FRS	RA	RB	XO	/
0					5	6 10	11 15	16 20	21 30	31

Table 532.

Table 533. Load/Store indexed instructions XO-field

Bit	Description
21	Store/*Load*
22	*Secondary (active low)*
23	*Cross*[7] (active low)
24	DP/*SP*
25	Update
26:30	01110

Table 533.

Table 534. FP2 Load Indexed instructions

Instruction	**mnemonic**	**XO**[8]	**description**
Load Floating-Point Parallel Double Indexed	lfpdx	462	DW[EA]-> T_P, DW[EA+8] -> T_S
Load Floating-Point Parallel Double Update Indexed	lfpdux	494	DW[EA]-> T_P, DW[EA+8] -> T_S
Load Floating-Point Parallel Single Indexed	lfpsx	398	W[EA] -> T_P W[EA+4] -> T_S
Load Floating-Point Parallel Single Update Indexed	lfpsux	430	W[EA] -> T_P W[EA+4] -> T_S
Load Floating-Point Secondary Double Indexed	lfsdx	206	DW[EA]-> T_S
Load Floating-Point Secondary Double Update Indexed	lfsdux	238	DW[EA]-> T_S
Load Floating-Point Secondary Single Indexed	lfssx	142	W[EA]-> T_S

[7]store parallel as integer word index = (store secondary cross SP) 10000

[8]not used: 78, 110, 14, 46

Load Floating-Point Secondary Single Update Indexed	lfssux	174	W[EA]-> T_S
Load Floating-Point Cross Double Indexed	lfxdx	334	DW[EA+8]-> T_P, DW[EA] -> T_S
Load Floating-Point Cross Double Update Indexed	lfxdux	366	DW[EA+8]-> T_P, DW[EA] -> T_S
Load Floating-Point Cross Single Indexed	lfxsx	270	W[EA+4]-> T_P, W[EA] -> T_S
Load Floating-Point Cross Single Update Indexed	lfxsux	302	W[EA+4]-> T_P, W[EA] -> T_S

Table 535. FP2 Store[9] Indexed instructions

Instruction	mnemonic	XO[10]	description
Store Floating-Point Parallel Double Indexed	stfpdx	974	S_P,S_S -> DW[EA],DW[EA+8]
Store Floating-Point Parallel Double Update Indexed	stfpdux	1006	S_P,S_S -> DW[EA],DW[EA+8]
Store Floating-Point Parallel Single Indexed	stfpsx	910	S_P,S_S -> W[EA],W[EA+4]
Store Floating-Point Parallel Single Update Indexed	stfpsux	942	S_P,S_S -> W[EA],W[EA+4]
Store Floating-Point Parallel as Integer Word Indexed	stfpiwx	526	S_P,S_S -> W[EA],W[EA+4]
Store Floating-Point Secondary Double Indexed	stfsdx	718	S_S -> DW[EA]
Store Floating-Point Secondary Double Update Indexed	stfsdux	750	S_S -> DW[EA]
Store Floating-Point Secondary Single Indexed	stfssx	654	S_S -> W[EA]
Store Floating-Point Secondary Single Update Indexed	stfssux	686	S_S -> W[EA]
Store Floating-Point Cross Double Indexed	stfxdx	846	S_P,S_S -> DW[EA+8],DW[EA]

[9]Store Single operations: mantissa truncated, Exponent overflow forced to infinity, sub-denormal underflow forced to zero.

[10]not used: 590, 622, 558

Store Floating-Point Cross Double Indexed Update	stfxdux	878	S_P,S_S -> DW[EA+8],DW[EA]
Store Floating-Point Cross Single Indexed	stfxsx	782	S_P,S_S -> W[EA+4],W[EA]
Store Floating-Point Cross Single Indexed Update	stfxsux	814	S_P,S_S -> W[EA+4],W[EA]